モダンな機能を搭載した標準プロトコル(IPsec / IKEv2)ベースの ユーザ空間で動作する VPN ソフトウェアの設計と実装

花田 哲治

Rockhopper プロジェクト (http://rockhopper.sourceforge.jp/)

Rockhopperプロジェクトでは、標準プロトコル IPsec(ESPv3/IKEv2)を採用し、かつ仮想イーサネット機能、ルーティングベース VPN、ロールベースのポリシー管理機能、Peer-to-Peer(P2P)型通信支援機能、モバイル環境およびマルチホーミング環境への対応(MOBIKE)などのモダンな VPN 機能を搭載した Linux 上で動作する VPN ソフトウェア実装をスクラッチから設計および開発している。伝統的な IPsec の設計モデルでは暗号プロトコル(ESP)の処理はデバイスドライバとして実装されてきたが、本実装ではトンネルインタフェース(TUN/TAP デバイスドライバ)を利用した近年一般的になってきたアプローチを採用、VPN 管理機能(IKEv2)を含め全てをユーザ空間で実装している。これによりソフトウェア構成がシンプルになるだけでなく、OS の保護下で全ての処理を安全に実行でき、管理や運用、開発やデバッグが容易になるなどの多くのメリットに与っている。またマルチコア CPU や暗号オフロード・ハードウェアをできるだけ活用しやすくするための設計も採用している。さらにベンダ依存度や運用システム依存度が高くカスタマイズが必要と考えられる各種機能を柔軟にカスタマイズ、プラグインできる構造となっている。そして、利用者による VPN の接続/切断や設定、モニタリングなどのために AJAX(Comet)ベースの Web 管理インタフェースを提供する。本プロジェクトの成果はオープンソース(LGPL ライセンス)として公開する。

1. Rockhopper プロジェクト：開発の背景と動機

近年、インターネット上での通信セキュリティに対する需要が高まるに連れ、よりユーザの視点にたって設計された様々な機能を持つ VPN ソフトウェア実装が提供されてきた。TCP/IP 環境においてセキュアな VPN を実現するための標準プロトコルといえば IPsec(ESP/AH および IKE(Internet Key Exchange))ということになる。しかし、実際には標準プロトコルではなく、独自またはプロプライエタリなプロトコルを採用している VPN ソフトウェアも多い。これには例えば IETF (Internet Engineering Task Force)における標準化プロセスの遅れやその非常に複雑な仕様により、開発者やユーザがそのメリットを理解しにくい状況が長く続いたことも影響している[B.Schneier]。実際、上述の高度な VPN ソフトウェアには、IPsec 実装に対する問題意識を持って実装されたものも存在する[OpenVPN][C.Hosner]。

しかし、相互接続性の観点やオープンな仕様を採用することでセキュリティプロトコルとしての安全性を公にコミットすることができる点、開発者間の容易な意思疎通を期待できる点など、標準プロトコルの採用が重要であることに依然として変わりはない。IPsec は次世代 IPv6 ネットワークにおいては標準機能として提供され、将来に渡ってもユーザに安定してかつ容易に入手可能な技術と考えられ、その意味でもその採用は最適である。

一方、複雑過ぎた IKEv1 仕様の反省を踏まえ仕様のシンプルさ、信頼性の確保、安全性の追求を目指しスクラッチから設計し直された IKEv2 プロトコルの標準化が完了し、また複数のオープンソース実装([racoon2][openikev2][ikev2prj][strongswan])により IKEv2 プロトコル自体の有効性が近年実証されてきた。また上述の IPsec 以外のプロトコルを利用した他の VPN ソフトウェアが蓄積してきた数々の経験により実際にユーザが必要とする機能についても明らかになってきた。例えば IP 上に仮想イーサーネットをオーバーレイする機能や P2P コミュニケーションを支援する機能などである。さらにこれらの VPN 実装および近年新たに出現してきているアプリケーションにより、伝統的な IPsec 実装モデルに対するいくつかの設計示唆を読み取ることもできる。

これらの経緯を踏まえ、既存の主なオープンソース実装とは異なった設計方針に基づく標準プロトコル(ESPv3/IKEv2)ベースの新たな実装の提供を目指し Rockhopper プロジェクトを始動した。

2.本ソフトウェアの設計方針と機能概要

既存の他の主要な IKEv2 オープンソース実装の中にも、[racoon2]などすばらしいソフトウェアが存在するが、本実装ではそれら他の実装とは異なるソフトウェア設計方針を採用している(暗号プロトコル(ESP)を含め全てをユーザ空間で実装、ルーティングベース VPN に主眼を置いた設計、セキュリティレベルに応じたプロセス分離、ESP 以外のトランスポート拡張(仮想イーサネット機能など)を統合、ロールベースのポリシー管理、設定・認証情報のデータ形式として Web 管理インタフェースにフレンドリな XML 仕様の採用など。詳細は後述)。そのため思い切ってスクラッチから実装する方法を選択した。また、本プロジェクトでは IKEv2 標準仕様の参照実装を提供することは目指していない。そのため広義の意味での「VPN」の観点から見て、利用シーンが実際に想定できるような機能や仕様にフォーカスして実装を行っていく方針である。IKEv2 の扱いに関しては、VPN 管理プロトコルまたオーバーレイネットワーク管理プロトコルとしてメリットのある場合、鍵管理以外の目的のためにも利用するという実利的な姿勢をとる。

IKEv2 仕様を規定する RFC4306 では(1) 拠点（サイト）間 VPN、(2) リモートアクセス VPN、(3) エンドポイント(Peer-to-Peer)間接続が IPsec の運用モデルとして想定されている。ここでエンドポイントとはサーバ、PC などを指す。もともと IPv6 の標準化プロセス内において(3)の Peer-to-Peer 通信のセキュリティを確保することも IPsec 仕様策定の重点の一つであったが、現実には主に(1)(2)の運用が主に普及している。例えばいくつものネットワーク機器ベンダが、これらの運用で利用しやすいゲートウェイ機器(GW)とクライアントソフトウェアを提供している。現状のビジネス観点ではゲートウェイ製品の提供が主眼であり、そのため GW 間((1)のモデル)または GW と専用 VPN クライアントソフトウェアとの接続((2)のモデル)がフォーカスされ、現状では(3)の運用はあまり普及していない。また(2)で提供されるクライアントソフトウェアは、多くのケースでオープンソースでは提供されていない。本実装では(1)(2)のモデルで利用できることをまず最初の目標としているが、最終的にはアプリケーション開発者やユーザにとって非常に有用なモデルである(3)でも利用しやすい方法の一つを提供することを目指す。その展望の詳細は後述する。このモデルではエンドポイントのアプリケーション開発プロセスに IPsec 機能を容易に組み込めるかが重要であり、(1)(2)モデルとは異なる有効な利用方式を模索する。

また、近年多くの VPN 実装で提供されるている TCP カプセリング(実装例:SSL-VPN、PPP over SSH など)については、もちろんファイアウォール装置に対して透過的な点や NAT 透過機能と相性が良い点(例:アドレスマッピングテーブルの無通信監視時間が TCP の場合長く管理される)など利点があるものの、以下の理由により多少保守的な立場を取

り現時点ではあまり重点をおいていない。

(1)トランスポートレイヤ(L4)から上位で提供されるトラフィック属性(再送制御やフロー制御など)が同一のデータグラムに対して、カプセリング前後の両方で重複して適用されることによる不整合問題(いわゆる「TCP over TCP」問題[TCPoTCP])。この問題を避けるため多くの実装では再送されたパケットを適当に間引くなどの処理を追加しているが、ネットワークの適切なレイヤー化という TCP/IP の基本コンセプトから得られる利点に反する部分もあり疑問が残る。

(2)通常の Web/SSH 通信などに見せかけて暗号通信がファイアウォールを通過できる点はネットワークセキュリティ管理者にとって本当に幸せなのかクリアでない。

2.1 柔軟'すぎない'設定:「複雑性は安全の敵」

IPsec 仕様では、非常に柔軟でかつ細かいセキュリティ・パラメータ設定を行えることが特徴として言われることがある。例えば暗号化方式、ハッシュ方式などのアルゴリズムの組み合わせや利用の有無を個々に選ぶことができる。しかし逆にこの特徴ゆえに管理者は非常に細部にわたる設定を意識的に行わなければいけないという見方もできる(2005年にはこの設定の複雑さに起因する脆弱性レポートが NISCC から報告されたことすらある[NISCC])。一方で別の代表的なセキュリティプロトコルである SSL/TLS プロトコル[RFC4346]の実装では、これらアルゴリズムの組み合わせを暗号スイートとしてあらかじめビルトインしておき、セキュリティ強度または暗号処理負荷のコスト的に最もリーズナブルなものをピア間でネゴシエーションするというアプローチをとっている。これらは実装思想の違いによるものであるが、本ソフトウェアでも十分に安全性に配慮しつつ、この柔軟すぎない設定を基本の設計思想として採用する。

2.2 本ソフトウェアの特徴

本ソフトウェア実装の主要な特徴は以下の通りである。

(1) 暗号プロトコル(ESPv3)処理を含めて VPN 管理プロトコル(IKEv2)処理とともに全てをユーザ空間で動作するアプリケーションとして実装。

(2) 認証処理やシステム情報へのアクセス処理といったセキュリティ的に保護すべき機能を別プロセス・ユーザ権限として動作可能に実装し、ネットワーク処理など外部とアクセスする機能から安全に隔離。

(3) 暗号オフロード・ハードウェアやマルチコア CPU といった近年利用可能になったハードウェア環境へ対応。

(4) ベンダ依存度または運用システム依存度の高いと考えられる機能に対し SPI(Service Provide Interface)を定義、外部ライブラリを利用したサービスプロバイダ実装としてプラグイン可能な構造。

(5) AJAX(Comet)ベースの Web 管理インタフェース。

(6) 仮想イーサネット機能、ルーティングベース VPN、P2P 型通信の支援機能など現在広く使われているモダンな VPN 機能を標準ベースの実装へ統合。

(7) ロールベースの IKEv2 認証、VPN レルム管理。

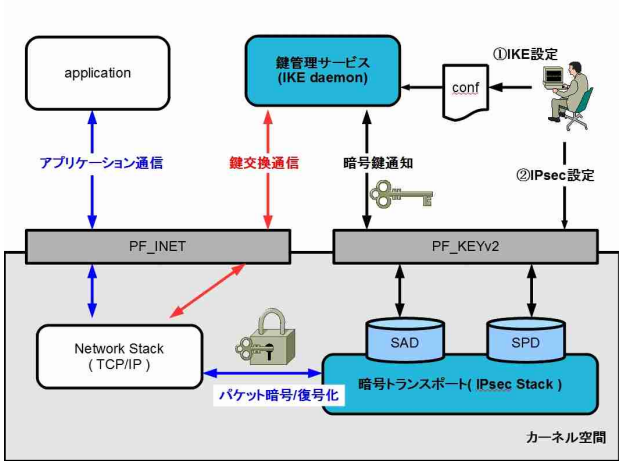
(8) モバイルネットワークおよびマルチホーミング環境の標準サポート(MOBIKE)。

(9) 拡張機能を後からプラガブルに追加できる構造。

本ソフトウェアでは、VPN またはオーバーレイネットワークに関連する拡張機能を手軽かつ柔軟に実装、デバッグおよび実験できる標準プロトコルベースの基盤ソフトウェアを提供することをまずは実現する。

2.3 IPsec/IKE の伝統的な実装モデル

伝統的な IPsec 実装では、例えば SSL/TLS プロトコルなど他のインバンド型のプロトコルとは異なり、鍵交換などの機能を持つ VPN 管理プロトコル(IKE)と暗号プロトコル(ESP/AH)が相互に独立な形で設計されている[RFC2367]。また後者の暗号プロトコル機能はほとんどの場合、オペレーティング内で動作するデバイスドライバとして提供されている。もちろん、この設計モデルは暗号プロトコルが特定の暗号鍵管理サービスにのみ依存することを避けるという意味で非常に汎用的な設計であり、かつトラフィックのシリアライゼーションとデシリアライゼーション処理がカーネル空間で実行されるため効率的かつ高速であるなど多くのメリットを持つ(前述の IKEv2 を採用した他の主なオープンソース実装は全てこのモデルに準じた実装である)。



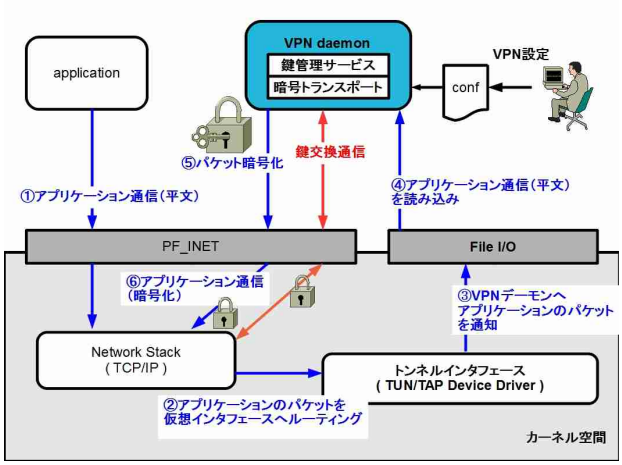
[伝統的な IPsec アーキテクチャ(図 1)]

一方で IPsec 機能を実装または利用する立場からすると、独立した2つの仕様およびその関係性を意識しなければならないという点において扱いが複雑になるという意見もある。例えば Linux などの IPsec 機能を利用する典型的な設定シナリオでは、IKE サービスと ESP/AH プロトコルスタックの設定エントリポイントが分離しているため、それぞれへ設定情報を送り込まなければならない。また機能拡張を行う場合において、IKE サービスと ESP/AH プロトコルスタックの密な連携を必要とすることも少なくはないが(例えば IKEv1 の Dead Peer Detection [RFC3706]で無通信監視機能を実装する場合など)、カーネル空間で動作するモジュールの開発はユーザ空間で動作するソフトウェアの開発に比べ、一般的にハードルが高いといわれることもある。例えば開発環境の充実度やデバッグのしやすさ、特権モードで動作するゆえのリソース管理、スケジューリング、セキュリティの確保の難しさ、オペレーティングシステム毎に異なる固有動作や実装作法と競合しない設計の必要などである。さらにこのように分離して設計する場合、IKE デーモンと IPsec デバイスドライバ間の API 仕様を規定する必要があるが、多くの実装で採用されている PF_KEYv2 API [RFC2367]は必ずしもシンプルとは言えない仕様である(例えば IPsec トンネルを接続する際の IKE デーモンと ESP/AH デバイスドライバ間でのかなり複雑な手順や同一トンネル接続を構成するアウトバウンド・インバウンド SA 管理を統合的に扱う API がないなど)。

2.4 すべてをユーザ空間で実装

近年人気を博している他の VPN ソフトウェアの中には、暗号プロトコルを IPsec の伝統的設計モデルのような分離した

機能としてかつデバイスドライバによって実装するのではなく、インバンド型でかつすべてユーザ空間で動作するアプリケーション内へ実装しているものも多い。これは最近の多くのオペレーティングシステムに備わっているトンネルインタフェース機能(TUN/TAP デバイスドライバ)[tuntap]を利用することで実現されている。(図 2)でその動作原理を簡単に述べている。



[全てをユーザ空間で実装するアーキテクチャ(図 2)]

このアプローチには、前述の IPsec/IKE の伝統的な実装モデルにはない様々なメリットがある。まずユーザにとっては、設定のエントリポイントが一つになるためその設定方法が理解しやすくなる点である。また開発者にとっては VPN ソフトウェアとしての様々な機能を実装する際にその開発作業の容易さも確保している。そしてこのようにユーザ空間アプリケーションとして動作するということにより、特権ユーザ以外の権限で実行できる点やケーパビリティを柔軟に制御できる点などオペレーティングシステムの保護下で安全に実行できることが何よりも注目すべきメリットである。特に近年の例えばセキュアOSのようなオペレーティングシステムによるセキュリティ機能が急速に発展しているトレンドの中で、ますます大きな恩恵を受けられる可能性がある。

IKEv1 仕様の汎用性のため導入された(そしてそれ故にその複雑な仕様の原因の一つだった)DOI (Domain Of Interpretation)の考え方が削除されたため、IKEv2 は事実上 IPsec 専用のプロトコルになっている(かつ実際の利用シーンでもほとんどのケースで IPsec (ESP)は IKE と共に運用されている)ため、IPsec 設計の汎用性という観点では管理プロトコルと暗号プロトコルを分離する意味が薄らいでいる。

これらを踏まえ、もちろん伝統的な IPsec 実装モデルにも多くの利点があるものの、本ソフトウェアで実装を行う機能の中には暗号プロトコル(ESP)機能部分にも影響するものも多く実装の容易さを確保する、かつオペレーティングシステムによるサポートを受けてセキュアな設計をすることができるという利点からこのユーザ空間での実装アプローチを採用している。ただし、現状の多くのオペレーティングシステムにおいて、トンネルインタフェースを利用する場合、同一パケットのコピーがユーザ空間とカーネル空間の間で何度も行われるため、そのオーバーヘッドによりある程度の性能を犠牲にしてしまう。この点については、トレードオフの問題でもあるが、今後とも継続してもちろん改善を考慮しなければならない。

2.5 仮想イーサネット (レイヤー 2VPN)

近年、IP レイヤ上に仮想的なイーサネットをオーバーレイする運用形態が多くの VPN ソフトウェアにより提供されている。この仮想イーサーネットは一般的に仮想ネットワークインタフェース(トンネルインタフェース)と仮想ハブの2種類のエンティティにより構成され、それらノード間が IP ネットワークにて接続されていても、あたかも仮想的なイーサネット環境で通信を行っているように見せることができる機能である。非常に多くのノードを抱えるようなネットワークではあまり効率的ではないが、一方で IP 接続できる環境であれば LAN 環境と同じ感覚で(例:IP ルーティングの設計・設定が不要など)、容易にネットワークを運用できる点は、小中規模のネットワークではメリットがある。

一般的に本機能はイーサーネットフレームを IP パケット上でカプセリングすることで実現する。本機能を実現するための標準プロトコルとして、高度に LAN 接続をエミュレートすることのできる Layer 2 Tunneling Protocol(L2TPv3)[RFC3931]があるが、本ソフトウェアではまずはイーサネットフレームの IP カプセリングのみを行うシンプルな仕様である EtherIP[RFC3378]を採用している。ただし、L2TPv3 と異なり EtherIP では細かいエミュレーションができないため以下の代替手段をいくつか講じる必要がある。

IPsec トンネル上でエミュレートされる仮想イーサネットの MTU 長は、IKE によってネゴシエーションされた暗号アルゴリズムの要求(IV 長やブロック長など)に依存して変化する(IPsec を利用する環境では、パケットのカプセリングを行うために送受信できるパケット長は一般に小さくなる)。つまり、イーサーネットのエミュレーションで行われるピアノード間での MTU 長の同期ネゴシエーションはこれで代用される。

またパケットシーケンス制御(送信した順序通り受信する)も同エミュレーションで要求されるが、IP ネットワークではパケット到達順序の入れ替わりが原理的に発生する。そのため ESP ヘッダのリプレイ攻撃防御用のシーケンス番号を利用したシーケンス制御を実装する。

本ソフトウェアでは仮想ハブ機能を実装し、MAC アドレス学習機能(ピアノードのトンネルインタフェース MAC アドレスと VPN 接続のマッピングテーブル管理機能)やブロードキャストや MAC 未学習時のパケットのフラッディング、ARP キャッシュ機能などを搭載している。

ARP キャッシュ機能とは、ARP 要求のフラッディングを抑制するためのものである。本機能を有効にすると、仮想ハブは、次に別ノードからの ARP 要求を受信した際に、学習済み MAC アドレスと IP アドレスのマッピングを参照して ARP 代理応答する(これはローカルのオペレーティングシステムのネットワークスタック経由で発生した APR に対しても同様に機能する。つまりローカルノードがブリッジ構成で設定されていても VPN 側に流れる ARP フラッディングが抑制される)。また IKE_AUTH 交換やその後の INFORMATIONAL 交換によって、ピアノード間でその MAC アドレスを交換し互いに MAC を学習する拡張も行っている(この場合には ARP によるアドレス解決は行われない)。今後は DHCP スヌーピングや IGMP スヌーピングなどのトラフィック量を抑えたり、セキュリティを高めるための拡張も行っていく計画である。

2.6 ルーティングベース VPN (レイヤー 3VPN)

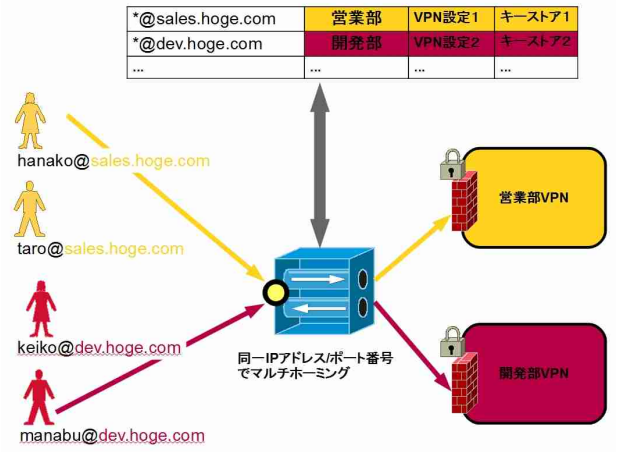
IPsec の基本アーキテクチャ[RFC4301]についてまず説明される機能の一つに「トラフィックセレクタ」ベースのアクセス制御がある。そこでは送信元・送信先 IP アドレスやプロトコルなどのフィルタ条件(セレクタ)で対象トラフィックの扱い方を指定する。この機能は IPsec の抽象的な概念モデル SPD(Security Policy Database)内の機能として説明され、これらフィルタ条件にマッチしたトラフィックに対して通過・破棄・暗号化の操作を適用することになっている。ただし実際

には暗号化操作の対象トラフィックを識別するために主眼が置かれる場合が多い（実装によっては IPsec によるフィルタ定義は粗すぎるため、通過・破棄の処理はステートフルインスペクション機能などのより高度かつ柔軟な機能を持ったファイアウォール機能へ任されることもある）。一方、最近の運用例では IPsec 専用のトンネルインタフェースを定義し、それを出力インタフェースとしてルーティングテーブルへ登録することで、ルーティング情報に従って IPsec 対象トラフィックを絞り込む実装を利用する場合も多い（この際セレクタとして全トラフィックを対象として指定し、IPsec トンネル毎の細かいセレクタ定義をしなくても IPsec VPN を運用できるようにする場合も多い。代わりにファイアウォール機能を利用する）。この場合一般的な IP ルーティング設定とまったく同じ考え方で VPN を運用できる点が最大のメリットである。

本ソフトウェアでは前述したように、トンネルインタフェースを利用した実装を行うためルーティングベースによる VPN コンフィグレーションも同時に可能となる。

2.7 ロールベースの VPN レルム(ポリシー)管理

本ソフトウェアでは、あるセキュリティポリシーを共有するノードを「VPN レルム」という単位へロール化して管理するモデルを採用している。



[ロールベースの VPN レルム管理(図 3)]

IKEv2 では IKE_AUTH 交換によってノードは互いに ID を交換することで相互認証を行うが、その際にそのピアノード ID が属するロールを判断し、それぞれのロール毎に用意された VPN 設定やキーストアへマッピングする。もちろん同一の IP アドレス/ポート番号上で複数のロールを定義することが可能であるため、マルチホーミング機能を実現できる。ピアノードの ID がどのロール（すなわちどの VPN レルム）に属するかは受信したその ID に対する部分マッチフィルタにより判断される。例えば部署毎に異なる E-MAIL アドレスのサブドメイン部や X.500 証明書 DN 内の RDN の組み合わせをロール識別条件として部署毎に定義し、それぞれ毎に異なる IKEv2/ESP 関連の設定、VPN 内部ネットワークの設定、認証方式、キーストアなどを指定することができる。

さらに IKEv2 自体の仕様でも、接続を行うピアノード側の ID をリモート端末が指定することができるマルチホーミング機能が追加された。この機能もロールベースのレルム識別に組み合わせて利用することができる。

2.8 モバイル環境を前提とした設計(MOBIKE)

近年の VPN の運用形態において、ワイヤレス LAN などモバイル端末からの VPN 接続が増えている。また、最近のラップトップ PC ではデフォルトで有線 LAN とワイヤレス LAN の両方が提供されている場合も多い。このような環境では、端末がアクセス・ネットワークを移動する度にアクセス情報がダイナミックに変わる場合も多く、IKEv1 仕様では基本的にスタティックな IP 接続のみを想定しているため効率的な運用ができない（IP アドレスが変更される毎に IPsec 接続を全てやり直す必要があり、リモートアクセス運用形態ではゲートウェイノードに負担がかかり、接続の復旧がリアルタイムに行えない場合さえある）。IKEv2 ではこのような運用のための MOBIKE （IKEv2 Mobility and Multihoming Protocol)[RFC4555]と呼ばれる機能が定義されている。本ソフトウェアでももちろんこの機能をサポートしている。

2.9 ネットワーク接続冗長化(マルチホーミング)

前述の MOBIKE 仕様を利用することで、IPsec 通信経路の冗長化(メイン・バックアップ経路構成、つまりネットワーク経路のマルチホーミング)も可能になる。本実装でもこの運用形態をサポートしている。

2.10 ユーザ端末のリモート・コンフィグレーション

最近のリモートアクセス型の VPN をサポートするほとんどのソフトウェア実装において、ゲートウェイ(またはコンセントレータ)ノードによるリモート端末への自動コンフィグレーション機能（例えば VPN 上でオーバーレイされるネットワークでの端末 IP アドレスの割り当て、DNS サーバの設定、ルーティング情報の設定など）がサポートされている。IKEv2 でもこの機能はもちろん仕様化されている。本ソフトウェアでも本機能をサポートする。さらに、例えばリモート端末へのバナーメッセージの送信機能など IKEv2 で定義されている以外の便利な拡張も行う予定である。

また、本ソフトウェアで実現される VPN 上では、レイヤー2（仮想イーサーネット）およびレイヤー3（ルーティングベース）におけるトラフィックが透過的になるため VPN 上で DHCP サービスを運用することもできる。

2.11 ブロードバンド・インターネット対応

ブロードバンド接続では例えば PPPoE による運用時のように、パケットの MTU 長が LAN 接続のものと異なる場合がある。さらに IPsec を利用する環境では前述したように送受信できる最大パケット長は一般に小さくなる。このためカプセル後に IP フラグメントが発生し、転送速度や品質が低下するなどの影響がでる場合がある。また、その長さは IKE によってネゴシエーションされた暗号アルゴリズムの要求(IV 長やブロック長など)に依存して変化する。このため本ソフトウェアでも一般的な VPN ソフトウェアと同様に、最適な MTU 長を自動的に制御する IPv4 の PMTUD(Path MTU Discovery)機能および TCP ヘッダの MSS の自動書き換え機能を実装する。

2.12 ハードウェア・オフロードを意識した設計

暗号化のための鍵生成、パケットの暗号・復号・ハッシュ処理は他に比べて、CPU に非常に高い負荷をかける処理である。そのため、近年ではこれらの処理を専用ハードウェアへオフロードするアプローチの採用も増えてきている。その実装方式はベンダにより様々ではあるが、一般的に以下の種類に分類できると思われる。

(1) 鍵生成・暗号/復号・ハッシュ処理のオフロード

(2) (1)に加え ESP/AH パケットのシリアライゼーション・デシリアライゼーションもオフロード

(3) (2)の機能を搭載した NIC でインライン型高速処理

本ソフトウェアでは、これらハードウェアへのインタフェース

を提供する外部ライブラリを柔軟にプラグインできる設計としている。ただし、デフォルトでは OpenSSL ライブラリ[openssl]を利用したソフトウェア実装のみを提供している。

2.13 マルチコア CPU を意識した設計

近年、マルチコア CPU を搭載した PC やサーバの採用が広がっている。本ソフトウェアでは、このマルチコア CPU を生かすためマルチスレッドによる実装を採用している。現状は、最も負荷の高い処理である暗号関連処理の分散を主眼においた設計としている。具体的には、ワーカースレッドプールを用意し、VPN トンネル毎にパケット処理などを、それぞれ適切な分散ルールに従って複数のスレッドへディスパッチする方式を実装している。また、暗号化・復号に必要なセキュリティ・パラメータ情報を TLS(Thread Local Storage)にキャッシュし競合を防ぐなどの試みも行っている。

2.14 外部ユーザ認証サービスとの連携

リモートアクセス接続のシナリオにおける多様なユーザ認証方式のため IKEv2 では EAP[RFC3748]をサポートしている。本ソフトウェアでは、他のサプリカント・ライブラリなど柔軟にプラグインできる設計としている。

2.15 外部 PKI ライブラリとの連携

VPN 接続をするノード間認証方式として PKI サービスを利用する運用も増えてきている。本ソフトウェアでは、他の X.509 証明書を処理するための PKI ライブラリを柔軟にプラグインできる設計としている。デフォルトでは OpenSSL ライブラリを利用した実装を提供している。

2.16 AJAX ベースの Web 管理インタフェース

本ソフトウェアでは、ユーザフレンドリな管理インタフェースを提供するために、Web(HTTP)/AJAX ベースのインタフェースをサポートしている。具体的には、本ソフトウェアの設定、VPN 接続/切断の操作、ユーザ認証、ログやモニタのユーザインタフェースとして Web ブラウザを利用できる設計としている。そのための通信方式として Comet[comet]と呼ばれる HTTP プロトコル上で双方向通信を実現するアイデアを採用している。HTTP による通信では通常サーバ側で発生したイベントを非同期にクライアントへ通知することはできないが、この Comet では、HTTP サーバがクライアントからの要求をイベント発生までフックし応答することで擬似的に双方向性能を実現する。本ソフトウェアでは、フルスペックな機能は必要ないため、専用のライトウェイトな Web サーバ機能(HTTP 1.0)を独自に実装している。また設定ファイルなどには Web ベース設計にフレンドリな XML フォーマットを採用している。

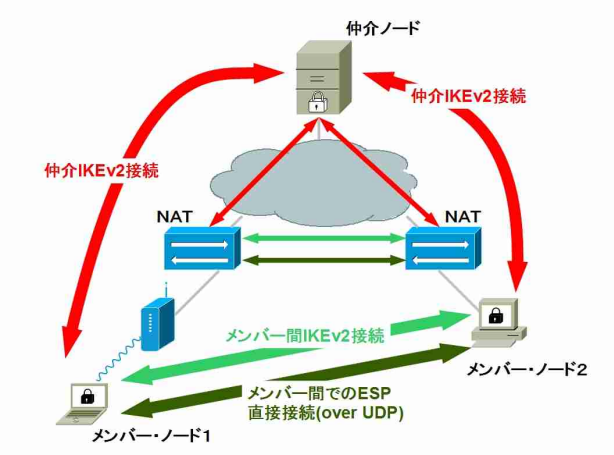
2.17 P2P 通信の支援

本ソフトウェアでは特定のセキュリティレルムへ属するノード間の P2P 型接続形態を支援する機能を搭載する。

本機能は IPsec の伝統的な観点から言えば、メッシュ型 VPN 接続の自動化を実現する機能である(IPsec の歴史的にはハブアンドスポーク型運用によるハブノードの負荷軽減や VPN 上で転送されるリアルタイムトラフィックの伝送遅延の軽減などの問題を解決するために発展してきた技術)。具体的には(1)ロケーションサービス(ピアノードの自動発見)、(2)NAT 透過機能を提供する。

(1)のロケーションサービス機能では、P2P 接続するピアノードへのアクセス情報を自動的に検出する機能を実装する。実装方式としては、インターネット側に配置されたその VPN レルムに属するメンバーノードのアクセス情報を管理する仲介ノードを用意し、他のノードがそれを問い合わせるという一般的な方式を採用する。

(2)の NAT 透過機能について、IKEv2 および ESPv3 では標準で UDP カプセリング機能が利用できるためそれを利用する。ここでは一般的な実装方式である UDP パンチホールディングやインターネット側へ配置された中継サーバを介して通信する方式などを同様に採用する予定である。(一部は他の IPsec 実装でも実現されている[ID-MEDIATION])。



[P2P 接続の概要(図 4)]

3. 実装の詳細と内部構造

ここでは本ソフトウェア実装の詳細や内部構造について特徴的な部分にフォーカスして紹介する。

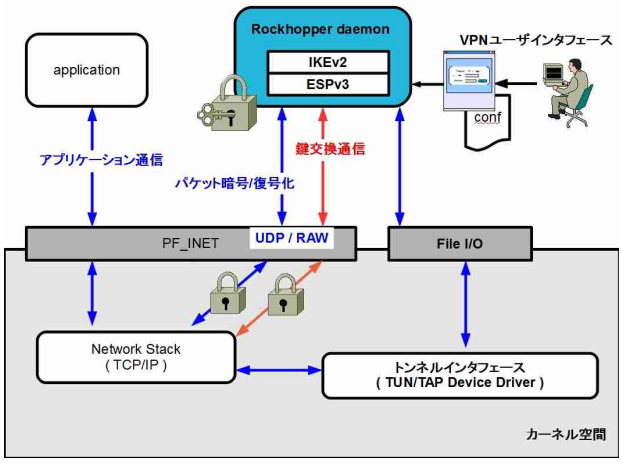
3.1 採用している標準仕様

用途	仕様
暗号(セキュリティ)プロトコル	ESPv3(RFC4301,4303,4835 など)
VPN 管理プロトコル	IKEv2(RFC4306,4307,4555,4718 など)
仮想イーサネット	EtherIP(RFC3378)
暗号アルゴリズム	AES,3DES,その他随時拡張予定
ハッシュアルゴリズム	SHA-1,MD5(後方互換のため),その他随時拡張予定
認証	X.509, EAP(RFC3748)
管理インタフェース	HTTP 1.0(RFC1945)、XML1.0

3.2 開発環境

Linux 2.6 ベース、実際には Ubuntu 9.04 上で開発している。他オペレーティングシステムへの移植は将来検討。開発言語はC言語、IDE には Eclipse 3.4 と各種プラグインを利用している。またまずは IPv4 のサポートに力を注いでいる。

3.3 アーキテクチャの概観

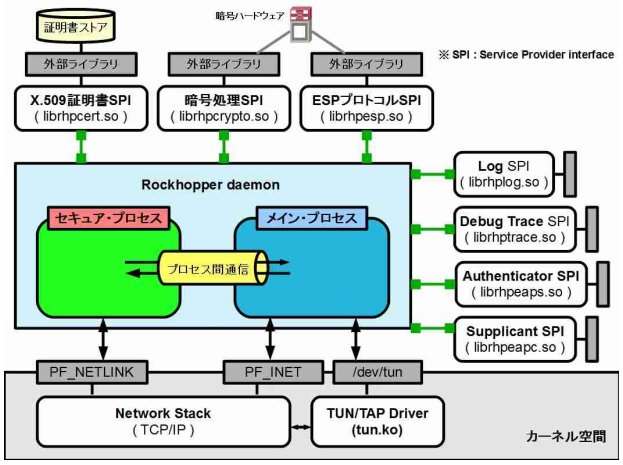


[システム上での構成(図 5)]

前述したように本ソフトウェアは図 2 で述べた他の VPN ソフトウェアですでに実績のあるアーキテクチャを採用し、動作するのはたった一つのユーザ空間アプリケーションのみというシンプルな構成である(図 5)。

一般にトンネルインタフェースといった場合、(1)ピアノードとの VPN 接続毎にトンネルインターフェースのインスタンスを 1 対 1 にマッピング(一般に Point-to-Point 接続と呼ばれる方式)、(2)VPN レルムに属する全てのピアノードとの VPN 接続に 1 つのトンネルインタフェースのインスタンスを共有する(一般に Multipoint 接続と呼ばれる方式)の 2 つの実装方式がある。本実装ではインスタンス数が少ない方が管理がしやすいという立場から後者(2)の方式を採用している。

また、本ソフトウェア動作時の外部ライブラリや外部サービスへの依存関係の概観は(図 6)の通り。ベンダ依存度、運用システム依存度の高い機能は SPI(Service Provider Interface)を実装した各種サービスプロバイダライブラリを経由して利用される。よってこのしくみを使って独自ライブラリを後からプラグイン可能である。

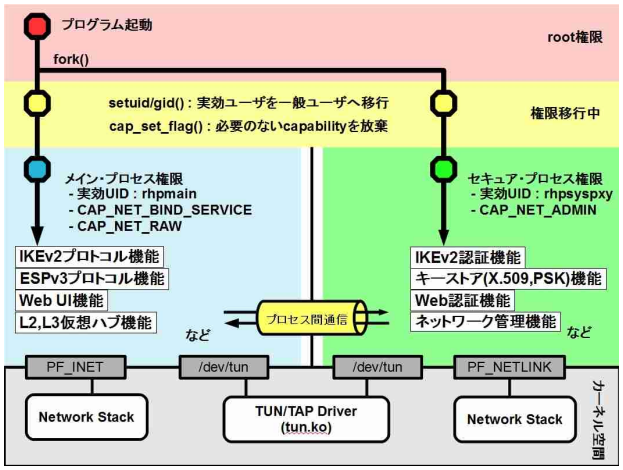


[他ライブラリおよびサービスへの依存関係(図 6)]

3.4 ライセンス

Lesser GPL(LGPL) v2.1 を採用している。

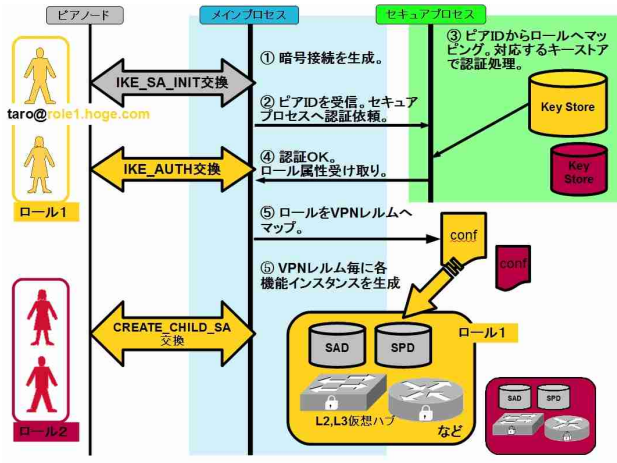
3.5 OS機能を最大限活用したセキュアな設計



[起動後のプロセス分離と状態遷移(図 7)]

本ソフトウェアではもちろんなるべくセキュアな実装を目指して設計されている。その一つとしてオペレーティングシステムのプロセスに備わっているセキュリティ機能や特性を利用した伝統的な設計方法に従っている。つまり、セキュリティ的に重要な処理を行う機能を別プロセスかつ別ユーザ権限で動作可能に実装し(ここではセキュアプロセスと呼ぶ)、外部とのアクセスを扱うネットワークプロトコル処理などを行う機能(メインプロセス)から安全に隔離する設計である(図 7)。

セキュアプロセスの隔離および実行ユーザ権限の移行が終了した後では、メインプロセスとのプロセス間コミュニケーションは、SO_PASSCRED ソケットオプションを指定した PF_UNIX ソケットによって渡されるクレデンシャル(プロセスの pid、uid)のチェックを受けたメッセージのみを使って行われる。また個々の機能に応じたセキュリティチェックがさらに行われる。(図 8)はロールベース IKEv2 認証を処理する手順。これら分離されたプロセスが協調して動作する例である。



[ロールベース IKEv2 認証シーケンス(図 8)]

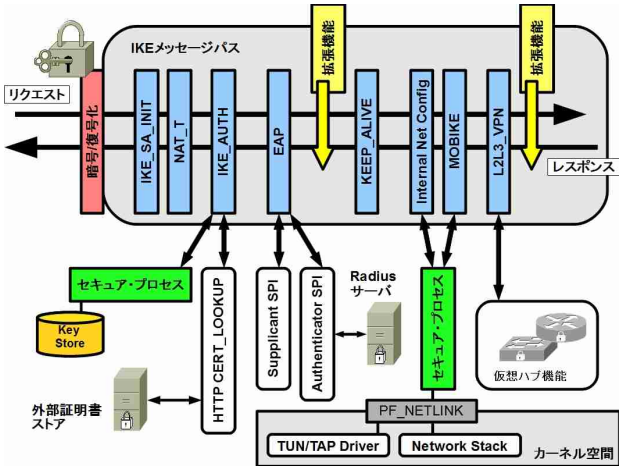
動作設定ファイルと認証情報ファイル(ロール定義、事前共有鍵(PSK)情報や X.509 証明書ストアへのパス情報、Web 認証情報などを記述)、個々のプロセスの実行ユーザは起動オプションで指定できる。

```
rockhopper -f conf_f -a auth_f -u u_name -s s_name
  -f conf_f :  動作設定ファイル名(XML)/パス (rhpmain 属性)
  -a auth_f :  認証情報ファイル名(XML)/パス (rhpsyspxy 属性)
  -u u_name :  メイン・プロセスの実行ユーザ名(rhpmain)
  -s s_name :  セキュア・プロセスの実行ユーザ名(rhpsyspxy)
```

[表 1 起動コマンドオプション]

3.6 拡張機能の追加を容易にするプラガブルな構造

新しい VPN 拡張機能を後から容易に追加できる設計とするために、Linux/netfilter のようなメッセージパスとメッセージハンドラによるプラグイン設計の考え方を IKEv2 プロトコル処理部に導入している(図 9)。



[IKE メッセージパスとプラグイン構造(図 9)]

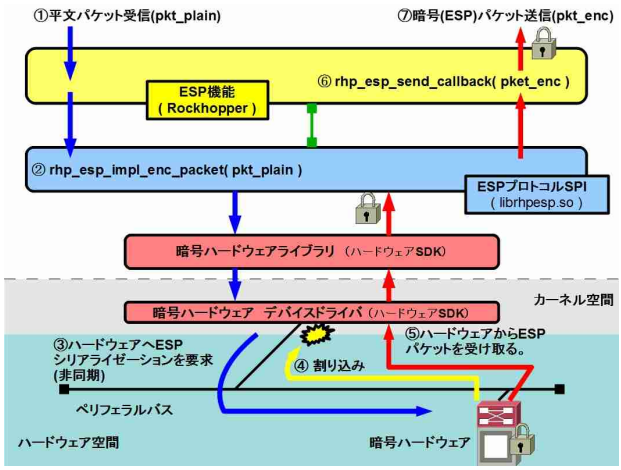
拡張機能はこのメッセージパス上の必要な場所へ表 2 の API を利用して後からプラグインすることができる。実際に IKEv2 サブ機能は全てこのしくみを利用して機能ごとに分離して実装されている(図 9 の青色部分)。また、拡張機能プラグインをライブラリとして実装した場合のため、それらを起動時にダイナミックにローディングする仕組みも計画している。

int rhp_ikev2_register_message_handler(int handler_type, RHP_IKEV2_MESG_HANDLER_TX_REQ send_request_mesg, RHP_IKEV2_MESG_HANDLER_RX_REQ_NO_VPN recv_request_mesg_no_vpn, RHP_IKEV2_MESG_HANDLER_RX_REQ recv_request_mesg, RHP_IKEV2_MESG_HANDLER_RX_RESP recv_response_mesg);	
handler_type	ハンドラがコールバックされる IKE メッセージパス上の位置を指定する。
send_request_mesg	IKEv2 要求メッセージの送信処理ハンドラ
recv_request_mesg_no_vpn	同上だが VPN が未確立状態の場合に使われる。
recv_request_mesg	IKEv2 要求メッセージの受信処理ハンドラ
recv_response_mesg	IKEv2 応答メッセージの受信処理ハンドラ

[表 2 IKEv2 メッセージハンドラ登録 API]

3.7 ハードウェアオフロード・ライブラリのプラグイン

近年、CPU 処理負荷の高い暗号処理部分をハードウェアオフロードすることができるようになってきた。ただし、多くの場合でそれぞれのハードウェアベンダが提供する異なる API やライブラリを使う必要がある。



[ESP 送信時のハードウェアオフロード処理シーケンスの例(図 10)]

本実装では(1)暗号プラグイン SPI(Service Provider Interface)と(2)ESP プラグイン SPI の2つの SPI を定義しそのサービスプロバイダをプラグインする設計となっている。(図 10)は ESP シリアライゼーションをハードウェアオフロードする処理シーケンスを示している。デフォルトではソフトウェア実装のみ提供している。

3.8 AJAX(Comet)ベースの Web 管理インタフェース

本実装ではユーザインタフェースを提供するために必要な最低減の機能のみを持つ小型の HTTP サーバを内蔵している。このサーバ機能は Comet ライク動作を想定しており、HTTP の TCP 接続毎に別ワーカースレッドを起動する設計ではなく、受信メッセージを処理する単一フロントエンド・スレッドとフックされるセッションを監視する単一バックグラウンド・スレッドが全てを順に処理するライトウェイトな設計となっている。またセキュリティ機能として Web 認証(現状は Basic 認証方式のみ)、および送信元 IP アドレス制限(デフォルトでは 127.0.0.1 からのみ許可)といった機能を実装している。

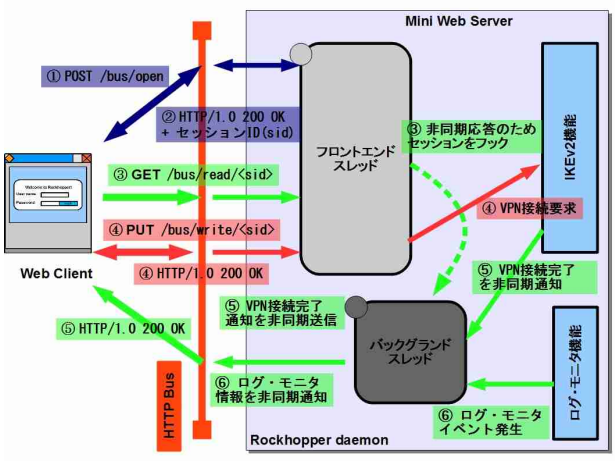
また「RHP Bus Protocol」と内部的に呼んでいる HTTP 1.0 上で動作し XML 1.0 仕様を Body メッセージとして利用する独自メッセージプロトコル仕様を定義している(表 3)(表 4)(図 11)。この仕様ではクライアントからのサーバへの操作モデルへ Bus I/O メタファを利用して open、write、read、close 操作を定義している。

HTTP リクエスト Body の XML 形式 [BusXmlReq]	<?xml version="1.0"?> <rhp_http_bus_request version="1.0" service="サービス名" ...> ... </ rhp_http_bus_request>
HTTP レスポンス Body の XML 形式 [BusXmlRep]	<?xml version="1.0"?> <rhp_http_bus_response version="1.0"> <rhp_http_bus_record index="0" service="サービス名" session_id="セッション ID" ...> ... </rhp_http_bus_record> <rhp_http_bus_record index="1" service="サービス名" session_id="セッション ID" .../> ... </rhp_http_bus_response>

[表 3 RHP Bus Protocol Body/XML メッセージ仕様の概要]

セッション操作	HTTP メッセージ形式(概要)
オープン(open()) Web 認証の終了後、サーバと Bus セッションをオープンする。応答としてそれ以降の操作セッションを識別する ID を受け取る。	[HTTP リクエスト] Header 部: POST /bus/open HTTP/1.1 Body 部: なし [HTTP レスポンス(成功時)] Header: HTTP/1.0 200 OK Body 部: [上述 BusXmlReq 形式] <?xml version="1.0"?> <rhp_http_bus_response version="1.0"> <rhp_http_bus_record index="0" service="new_session" session_id="セッション ID"/> </rhp_http_bus_response>
書き込み(write()) メッセージを Bus へ書き込む。応答メッセージを受け取ることも可能。	[HTTP リクエスト] Header 部: PUT /bus/write/<セッション ID> HTTP/1.1 Body 部: [上述 BusXmlReq 形式] [HTTP レスポンス(成功時)] Header 部: HTTP/1.0 200 OK Body 部: なし、または[上述 BusXmlRep 形式]
読み込み(read()) メッセージを Bus から非同期に読み込む。	[HTTP リクエスト] Header 部: GET /bus/read/<セッション ID> HTTP/1.1 Body 部: [上述 BusXmlReq 形式] [HTTP レスポンス(成功時)] Header 部:HTTP/1.0 200 OK Connection: Keep-Alive ... Body 部: [上述 BusXmlRep 形式] [HTTP レスポンス(サーバ側イベント待ちタイムアウト時)] Header 部: HTTP/1.0 404 Not Found
クローズ(close()) Bus セッションをクローズする。	[HTTP リクエスト] Header 部: DELETE /bus/write/<セッション ID> HTTP/1.1 Body 部: なし [HTTP レスポンス(成功時)] Header 部: HTTP/1.0 200 OK Body 部: なし

[表 4 RHP Bus Protocol HTTP メッセージ仕様の概要]



[RHP Bus プロトコルシーケンス例(VPN 接続・ログ・モニタ)(図 11)]

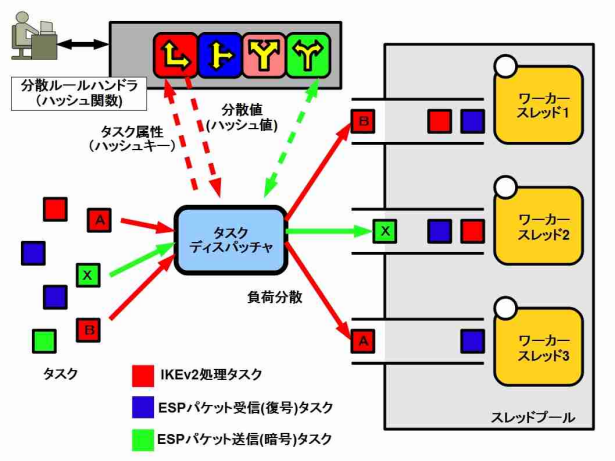
現状はテストプログラムによるデバッグ中であり、実際のメジャーな Web ブラウザ製品からのテストは未実施のため、

今後それらの動作に合わせて仕様のブラッシュアップが必要な場合には順次行っていく。

3.9 マルチコア CPU 環境への対応

3.9.1 スレッドプールを利用したルールベースのタスク負荷分散機構

マルチコア CPU 環境における効果的な処理実行のために本ソフトウェアではスレッドプールを利用したルールベースのタスク負荷分散機構を実装している(図 12)。分散処理はタスクディスパッチャと呼ばれる機能により実行される。スレッドプールを利用する機能は、まず自身のタスクのための負荷分散ルールを定義したルールハンドラを登録する。その後で各機能はタスクディスパッチャへタスク負荷分散を要求する。タスクディスパッチャはそのハンドラにより返される値によってプール内の分散先ワーカースレッドを決定しタスクをキューする。その後、ワーカースレッドは順次タスクを処理する。ルールハンドラは一種のハッシュ関数であり、どのようなアルゴリズムで処理タスクを分散するかのルールを自由に利用機能が定義するすることができる。例えば受信した ESP パケットの復号タスクを分散するために ESP ヘッダの SPI 値をハッシュキーとする、平文パケットの ESP パケットへのシリアライズと暗号化処理のためには送信先 IP アドレスをハッシュキーとする、特定の処理は必ず同一のスレッドへ割り振るなど自由に対応できる。



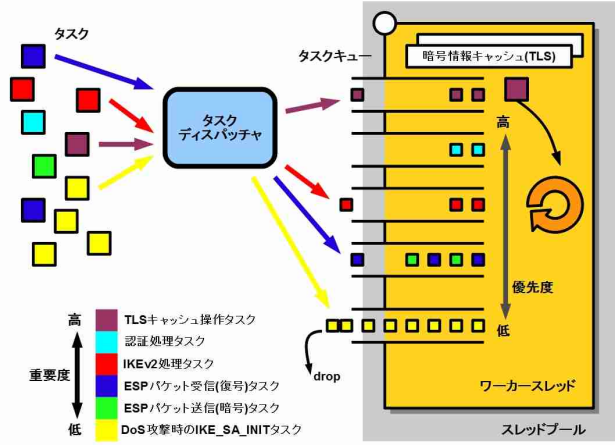
[ルールベースのタスク負荷分散機構の概要(図 12)]

このしくみにより後から運用環境やハードウェア実行環境に応じパフォーマンスチューニングが必要になった場合、起動ワーカースレッド数を調整したり、そのセマンティクスの変更に応じたアルゴリズムをルールハンドラへ実装することで比較的に自由度の高いカスタマイズを行えるような構造となっている。

3.9.2 タスク実行の優先制御機能

大量の暗号トラフィック(ESP)が発生した場合にその処理に CPU リソース割り当てが追いつかず、結果的に VPN 管理プロトコル(IKEv2)のトラフィックがなかなか処理されないことがある。これは伝統的な IPsec の実装モデルにおいても時折見られる現象である(実装に依存。例えば Linux では NAPI などにより軽減されてきたが ESP 処理はソフト割り込みコンテキストで負荷の高い暗号処理を行うことも一因か)。本来であればそれら管理プロトコル処理は優先的に処理されなければならない。

このような問題に対処するため本ソフトウェアではプール内の各ワーカースレッドは実行優先度毎にタスクキューを持っている(図 13)。利用者により指示された優先度に対応するタスクキューにタスクディスパッチャがタスクをキューイングする。



[タスク実行の優先実行機構の概要(図 13)]

また、この機構は IKEv2 上での DoS 攻撃の可能性を検出した場合にも利用されている。IKEv2 ではセッション開始時に攻撃者がハーフオープンの中途半端なネゴシエーション状態をレスポンダ側へ残す類の攻撃(TCP SYN フラッド攻撃と同じ手口)を緩和するしくみを持っている。これは IKEv2 の IKE_SA_INIT 交換時、レスポンダ側でステートレスクッキーをイニシエータへ返すことで実現される(TCP SYN Cookies と基本的に同じ考え方)。本実装ではハーフオープン状態のセッション数が指定された閾値を上回った場合、この機能が動作を開始する。この機能は専用のスレッド内でかつ実行優先度が最も低く実行され、DoS 攻撃の可能性がある場合に新規コネクション処理の実行頻度を制限することで、確立済みの他の VPN 接続処理への影響をできるだけ緩和する。

3.9.3 TLS(Thread Local Storage)を利用する試み

前述 ESP プロトコル SPI のサービスプロバイダ実装はデフォルトでは、ソフトウェア実装のみ提供されるが、ここでもマルチスレッド機能を有効に利用する試みを行っている。

ESP 処理のために現状デフォルトで組み込まれるタスク負荷分散ルールでは、パフォーマンスの観点から、復号(ESP 受信処理)と暗号化(ESP 送信処理)で別ワーカースレッドへディスパッチされるようになっている。そのため暗号鍵を管理している内部テーブルオブジェクトへのアクセス時に排他処理が行われる。しかし、暗号処理では CPU 時間を比較的長く費やすため、そのままではこのオブジェクトへのロックが一つのスレッドに長く保持され、他のスレッドは待機状態になってしまう。そのため暗号鍵情報などをワーカースレッド毎に TLS へキャッシュし、その後はロックなしで復号・暗号化処理を実行できるような工夫をしている。なお、TLS にキャッシュされた情報をクリアしたり更新したりする操作タスクは(図 13)にあるように最優先で各ワーカータスクにより実行され遅延をなるべく防いでいる。

4. 開発状況

現状では基本機能の実装作業はほぼ完了しておりテスト作業を進めている段階である。一方でいくつか未実装の機能についても並行して作業を進めている。

本ソフトウェアは SOURCEFORGE.JP において公開を予定している (http://rockhopper.sourceforge.jp/)。そのサポートに感謝の意を表したい。開発中のソースコードへのアクセス

は随時可能だが、一般向け v1.0 の公開は順調に行けば 2009 年末から来春くらいを予定している。

5. 今後の展望

5.1 Endpoint-to-Endpoint (Peer-to-Peer) 接続

冒頭に述べたように、現状 IPsec はエンドポイント（サーバ、PC など）による Peer-to-Peer のセキュア通信を行うために普及しているとは残念ながら言えない。エンドポイントでの IPsec 利用を想定できる状況には、その上で動作するアプリケーションが IPsec 機能を利用するケースを想定できる。アプリケーション開発者の観点からすれば、エンドポイントでは様々なアプリケーションが動作するため、そのセキュリティ要件や設定、認証情報もアプリケーション毎にそれぞれ異なる場合も多いだろう。伝統的な IPsec 実装の設計ではネットワークレイヤを中心に置いた設計がされているため、例えば「このユーザプロセス」や「このアプリケーション」のトラフィックといったより抽象的な指定による制御が得意ではない。例えばアプリケーション別のパスワードや X.509 証明書(例:異なる CA から発行された証明書)を利用する、同一アプリケーション間通信で TCP と UDP の両方を利用しかつそのポート番号はダイナミックに変化する、通常は平文で通信を行い、重要な情報をやりとりする場合だけ暗号通信を行うなどの場合に対応することが難しい。これはアプリケーション側から VPN の管理に必要な操作を行う手段が多くの場合提供されていないためである。だが本来であれば IPsec は下位レイヤで動作するため、アプリケーションからのトラフィックに対して整合的に動作可能な特徴を持っているはずである。しかし、セキュリティ管理の小回りが利かない現状は開発者に恩恵をもたらしているとは言えない。そのため現状では SSL/TLS や DTLS[RFC4347]を利用、または独自のセキュリティレイヤを実装することが一般的だろう。この方法が適している場合も多いが、以下のような理由を考慮すると IPsec の利用も有効な選択肢になりうるだろう。

(1)トランスポートレイヤ(L4)を含めて保護したい場合(近年続々と TCP プロトコル仕様または実装の脆弱性（例:TCP Session Hijack)[tcpvul]が報告されている)。

(2)同一アプリケーションで複数トランスポートプロトコルセッションを利用する場合(例:SIP)。この場合別々のセキュリティプロトコルを使うか、またはそのアプリケーション独自のセキュリティレイヤを導入しなければならない。これはそのアプリケーションの実装が煩雑になる可能性だけでなく、他の実装との互換性の再検証などの問題を生む場合がある。

(3) ダイナミックネットワーク上での運用。例えばモバイル環境においてクライアント端末が異なるアクセスネットワークへ移動した場合、割り当てられる IP アドレスが変更される場合がある。その際 TCP 接続はリセットされる。一方、IPsec トンネル上でオーバーレイされる動作（IPsec トンネルモード）を選択した場合、その接続には仮想トンネルインタフェースに割り当てられた内部 IP アドレスが利用されるが、これは VPN 接続が有効な限り変更はない。つまり TCP 接続や IP アドレスを意識するアプリケーションが VPN 上で通信を行っている場合にはコネクションまたはアプリケーション・セッションをリセットする必要がない。さらに内部ネットワークでのトラフィックに対する送信元のアクセス制限を管理する場合にも役立つだろう。もちろん Mobile IP のような汎用性や柔軟性はないが、MOBIKE 仕様が想定しているような、例えばクライアント-サーバモデルのような片方のピアのロケーションが代わらないような環境では有効かもしれない。

(4)IPsec は IPv6 規約上では標準搭載とされ、今後とも多くの OS で標準サポートされる可能性が非常に高い。

実のところ、この問題に対する一つのアイデアを提供することが、本プロジェクトの発端となった大きな動機の一つであった。これは次世代 IPv6 のネットワークでは IPsec 機能は全 TCP/IP 実装において、ほぼ標準搭載となると予定されているためアプリケーション開発者に対しても有用なソリューションの一つを提供できたらという思いが底流にある。このため以下のような技術的方向を模索している。

(A)IKEv2 のマルチホーミング機能を利用する。つまり、アプリケーションが IPsec 通信用のユニークなアクセスポイント ID をそれぞれ割り当てることで、同一の IPsec/IKev2 の IP/Port 上であってもアプリケーション毎に IPsec 通信を利用可能になる。本実装の文脈では、前述のレルムまたはロールをアプリケーション毎に割り当てることになる。この ID には、何らかの標準的な命名規約が必要かもしれない。

(B)IPsec/IKE サービスに対してアプリケーションが VPN 接続の開始や終了、サスペンドやレジュームを制御する API が必要だろう。

(C)アプリケーション毎に異なる可能性がある認証情報を IKEv2 サービスがアプリケーションとやりとりする API が必要であろう。

(D)アプリケーション毎のトラフィックセレクタなどの設定を IKEv2 サービスとアプリケーションがやりとりする API が必要だろう。ダイナミックにネゴシエーションされる複数のトランスポートプロトコルセッションを利用したい、またアプリケーションがダイナミックに暗号セッションのトラフィック条件やタイミングを制御したい場合には、セレクタを IPsec トンネル確立後であっても、アップデートできるしくみが必要かもしれない。

(E)できればマルチプラットフォーム(OS)間で共通に利用できるような IPsec 機能用の標準API仕様があると良いだろう。

(F)IPsec を採用している性格上、実際のカプセリング処理はアプリケーショントラフィックがトランスポートレイヤ(TCP/UDP など)の下位で行われる。アプリケーション毎のトラフィックを識別し適切な VPN へディスパッチするしくみを開発する必要があるかもしれない。

5.2 トンネルレス型 VPN

近年、トンネルレス型 VPN と呼ばれる新しい形態の VPN 運用が登場している。これはある特定のセキュリティレルムに属するノード間が、それぞれ個別の暗号鍵などのセキュリティパラメータ、つまり Security Association (SA)を生成する代わりに、キーサーバと呼ばれるノードがその他のノードへセキュリティ・パラメータを配布する。この機能を利用すると任意のノードと通信する場合でも、VPN トンネルの折衝をそれぞれ毎行う必要なしに暗号通信できるため、管理が容易でまた動作が軽いというメリットがある。本来マルチキャスト・トラフィックのための仕様として策定された IKEv1 の拡張仕様である GDOI(Group Domain Of Interpretation)を応用した実装[getvpn]が知られており、本実装でも同様の機能を実装するよう計画中である。

5.3 標準 IPsec 拡張機能への対応

現在 IETF の ipsecme(IP Security Maintenance and Extensions)ワーキンググループにて検討が進められている以下の標準仕様ももちろん実装していく計画である。

5.3.1 “Redirect Mechanism for IKEv2”

“Redirect Mechanism for IKEv2”[ID-REDIRECT]という拡張仕様は、ピアノードからの IPsec 接続を他ノードへリダイレクトするものである。シンプルな機能であるが、(1)ノードの

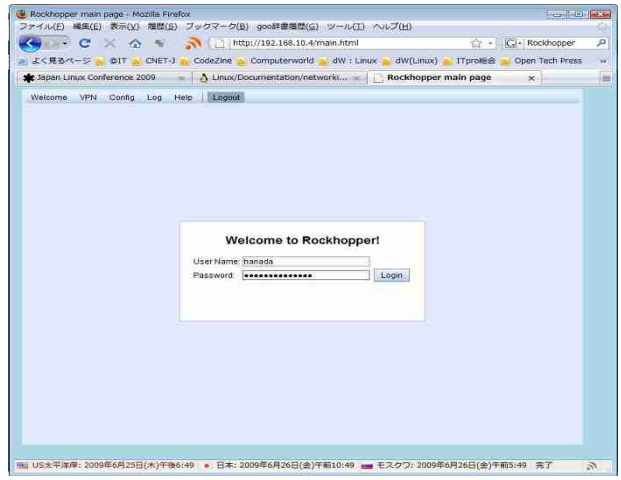
シャットダウン時に接続中の IPsec 接続を他ノードへ誘導する、(2)IPsec 接続を複数ノード間で負荷分散する、といった用途に利用できる。

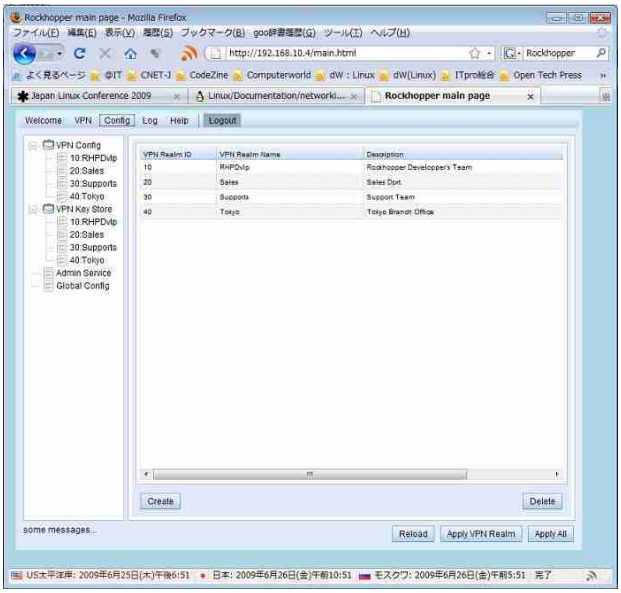
5.3.2 “IKEv2 Session Resumption”

現状の IKEv2 仕様では VPN 接続を何らかの理由（例:PC をハイバネーションする）でサスペンドする必要のある場合、次にその接続を再開する際、再び最初から IKEv2 ネゴシエーションをやり直す必要がある。これは特にリモートアクセス VPN 環境ではゲートウェイ（コンセントレータ）ノードに負荷をかける場合がある（EAP を利用してユーザ認証を行う場合には再度何度もピアノード間で IKEv2 メッセージ交換を行う必要があり、かつオーセンティケータおよび外部認証サービスにも負荷をかけることになる）。現在検討されている“IKEv2 Session Resumption”[ID-RESUMPTION]拡張仕様では、最初の IKEv2 認証完了時にそれを証明する有効期限のあるチケットをゲートウェイ（コンセントレータ）ノードがピアノードに対して発行することで、IPsec 接続のレジューム時に認証手順を簡略化することができる。

5.4 Web GUI のサンプル実装を提供

AJAX ベースで提供される Web 管理インタフェースを利用し、Web ブラウザで動作する GUI を自由に利用者がデザインできるが、Dojo Toolkit[dojo]を利用したサンプルも開発している。





[ブラウザ画面スナップショット(ログイン、VPN 設定画面)][開発中]]

5.5 CPU/プロセッサ アフィニティ

4つ以上のコアを持つ CPU を搭載したサーバや PC を安価に利用できる状況が近い将来やってくるかもしれない。将来的には前述したワーカースレッドプール内のスレッドをサブグループ化し、それらを VPN レルム毎に専用に割り当て、かつ特定の CPU コアにくくりつけることで VPN レルム毎の性能を保証する、DoS 攻撃を受けている最中に処理する CPU コアを特定のものへくくりつけるなどの工夫も現実的になるかもしれない。

6. まとめ

本オープンソース・プロジェクトでは、標準プロトコルである IPsec(ESPv3/IKEv2)の特徴を生かした新たな VPN ソフトウェア実装を提供する。これまで独自またはプロライエタリ・プロトコルで実装されてきた他の高度な VPN ソフトウェアにより培われてきた価値の高いモダンな機能を標準ベースのプロトコル上での実装へ統合することをまずは目指している。そして何よりも来るべく IPv6 イネーブルなネットワークにおいてデフォルト提供される IPsec 機能をアプリケーションの開発にも利用できるような方向を模索していきたい。

まだ始動して間もないプロジェクトであるが、実運用に耐えられるソフトウェア品質の早期確保を含め引き続き努力していきたい。

7. 参考文献・資料

・[B.Schneier] N.Ferguson, B.Schneier 2003 ”A Cryptographic Evaluation of IPsec” (http://www.schneier.com/paper-ipsec.html)
・[OpenVPN] James Yonan 2003 ”The User-Space VPN and OpenVPN” (http://openvpn.net/papers/BLUG-talk/index.html)
・[C.Hosner] Charlie Hosner 2004 ”OpenVPN and the SSL VPN Resolution” (http://www.sans.org/rr/whitepapers/vpns/1459.php)
・[pipsec] “pipsec” (http://perso.telecom-paristech.fr/~beyssac/pipsec/)
・[racoon2] Racoon2 (http://www.racoon2.wide.ad.jp/w/?Racoon2)
・[openikev2] OpenIKEv2 (http://openikev2.sourceforge.net/)
・[ikev2prj] IKEv2 Project (http://ikev2.zemris.fer.hr/)
・[strongswan] strongSwan (http://www.strongswan.org/)
・[NISCC] NISCC-004033 (http://jvn.jp/niscc/NISCC-004033/index.html)
・[TCPoTCP] Olaf Titz 2001“Why TCP Over TCP Is A Bad Idea” (http://sites.inka.de/~bigred/devel/tcp-tcp.html)
・[RFC4346] T,Dierks, E.Rescorla “The Transport Layer Security(TLS) Protocol Version 1.1”
・[RFC3706] G. Huang , et al. “A Traffic-Based Method of Detecting Dead Internet Key Exchange (IKE) Peers”
・[RFC2367] D. McDonald, et al. “PF_KEY Key Management API, Version 2”
・[tuntap] M.Krasnyansky, et al. “Universal TUN/TAP device driver.” (http://www.linux-m32r.org/lxr/http/source/Documentation/networking/tuntap.txt)
・[RFC3931] Lau, J, et al. ”Layer Two Tunneling Protocol – Version 3 (L2TPv3)”
・[RFC3378] R. Housley, et al. “EtherIP: Tunneling Ethernet Frames in IP Datagrams”
・[RFC4555] P. Eronen, Ed. “IKEv2 Mobility and Multihoming Protocol (MOBIKE)”
・[openssl] OpenSSL Project (http://www.openssl.org/)
・[RFC4301] S. Kent, et al. “Security Architecture for the Internet Protocol”
・[RFC3748] B. Aboba, et al. “Extensible Authentication Protocol (EAP)”
・[comet] “Comet (programming)” (http://en.wikipedia.org/wiki/Comet_(programming))
・[ID-MEDIATION] T. Brunner, et al. “IKEv2 Mediation Extension” (EXPIRED) (http://ietfreport.isoc.org/idref/draft-brunner-ikev2-mediation/)
・[dojo] Dojo Toolkit (http://dojotoolkit.org/)
・[tcpvul] 独立行政法人 情報処理推進機構セキュリティセンター 2009 年 “TCP/IP に変係る既知の脆弱性に関する調査報告書 改訂４版” (http://www.ipa.go.jp/security/vuln/vuln_TCPIP.html)
・[rfc4347] E. Rescorla, et al. “Datagram Transport Layer Security”
・[rfc3711] M. Baugher, et al. “The Secure Real-time Transport Protocol (SRTP)”
・[getvpn] ”Cisco Group Encrypted Transport VPN” (http://www.cisco.com/en/US/docs/ios/12_4t/12_4t11/htgetvpn.html)
・[labelnet] P.Moore “Introduction to Labeled Networking on Linux” (http://www.linuxfoundation.jp/jp_uploads/seminar20080709/paul_moore-r1.pdf)
・[ID-REDIRECT] V. Devarapalli, et al. “Redirect Mechanism for IKEv2 ” (http://www.ietf.org/internet-drafts/draft-ietf-ipsecme-ikev2-redirect-11.txt)
・[ID-RESUMPTION] Y. Sheffer, et al. “IKEv2 Session Resumption ” (http://www.ietf.org/internet-drafts/draft-ietf-ipsecme-ikev2-resumption-06.txt)